# Incremental Relational Fuzzy Subtractive Clustering for Dynamic Web Usage Profiling

Bhushan Shankar Suryavanshi
Dept. of Computer Science and Software Engineering
Concordia University
Montreal, Canada
bs_surya@cse.concordia.ca

Nematollaah Shiri
Dept. of Computer Science and Software Engineering
Concordia University
Montreal, Canada
shiri@cse.concordia.ca

Sudhir P. Mudur
Dept. of Computer Science and Software Engineering
Concordia University
Montreal, Canada
mudur@cse.concordia.ca

## ABSTRACT

A primary application of web usage profiling is in model-based collaborative filtering (CF) for building recommender systems used for web personalization. CF techniques used for recommendation require accumulation of vast amount of historical user-preference information, which is queried to provide a personalized experience. Model-based CF techniques are preferred over the somewhat more accurate memory-based CF techniques primarily due to their higher efficiency and scalability. In the situation where user interests change dynamically with time, memory-based CF allows easy addition of new usage data, but model-based CF often requires complete remodeling operation. Being computationally intensive, remodeling is done occasionally and the models used normally lag behind the current usage patterns leading to irrelevant and mistargeted recommendations. Even though development of maintenance schemes which adapt these models to non-stationary environments is very important, it has received less attention so far. Our first contribution in this paper is a web usage profile maintenance scheme using a new algorithm called incremental Relational Fuzzy Subtractive Clustering (RFSC). Incremental RFSC can efficiently add new usage data to an existing model overcoming the expense associated with frequent remodeling. We validate the results by showing close similarity between complete reclustering and the clustering models obtained after applying our incremental RFSC technique. Any maintenance scheme based on incremental update of the profile requires a measure to indicate, to the web analyst, when accumulated usage data has to be reclustered; otherwise continued maintenance leads to irrelevant, obsolete model. The second contribution of this paper is thus introduction of a quantitative measure, called *impact factor*. When the impact factor exceeds a predefined threshold, a remodeling is recommended. We conducted extensive experiments which compare the effectiveness of recommendations using our incremental RFSC technique, complete reclustering, and memory-based CF techniques. The results obtained indicate that our maintenance technique is almost as good as complete remodeling.

## Categories and Subject Descriptors
I.5.3 [**Pattern Recognition**]: Clustering.

## General Terms
Algorithms.

## Keywords
Collaborative filtering, dynamic profiling, clustering, web personalization, incremental RFSC.

## 1. INTRODUCTION

The World Wide Web is a huge information repository and an ever growing, most visited market place for online services and products. A vast variety of products and services available through the Internet has both simplified and complicated shopping online. While ordering has been rendered more convenient, selection has become more difficult for many by the sheer number of choices. Users often face the problem of information overload wherein the amount of information is far more than what can be easily assimilated and interpreted by human beings. Users generally feel lost in this huge information space. Web personalization is an approach to address this problem by providing content and services at a web site tailored to the needs of individual users from the knowledge gained through previous interactions of users with the site. Its goal is to improve the user's experience of an e-service. For example, in e-commerce, personalization can be perceived as an online salesman. A salesman understands the needs of a customer through the initial interaction with the customer. Recognizing and satisfying the user needs successfully will surely lead to a long lasting relationship and re-use of the services provided. Personalization can be compared to having your favorite bookseller pull out a copy of a book he just knows you would really like.

Profiles refer to information about individuals. Usage profiles capture different interests and trends among users accessing the site. These usage profiles can be used in personalization for recommendations, path prediction, prefetching of pages, better structuring of web sites, and in a nutshell improving the experience of the user. Usage profile information can also be used by content creators to understand which material is used more, how long a material is viewed by which types of users, and in what order material is accessed. The set of profiles reflects the semantics of the user historic dataset at particular point of time. Clustering is a widely used technique for discovering usage profiles from web log records [9, 11].

Web personalization process can be divided into four distinct phases [10, 11], namely collection of web data, preprocessing of web data, analysis of web data, and recommendation which makes use of the previous phases to provide recommendations to the user, such as adding hyperlinks to the last web page requested by the user, depending on the type of user. Recommender systems could be based on content-based filtering, collaborative filtering, or rule-based filtering [11]. Collaborative filtering (CF) is the most successful and widely used technique in building recommender systems [16]. The goal of CF is to predict the preferences of a user, referred to as the *active user*, based on the preferences of a group of "like-minded" users. The key idea in CF is that the active user will prefer those items that like-minded individuals prefer or even those items that dissimilar individuals do not prefer. This approach relies on history, a dataset recording all previous users' interests, which could be inferred from their ratings of the items at a website (products or web pages). Rating can be *explicit,* such as previous purchases, customer satisfaction questionnaires, etc., or can be *implicit*, such as browsing activities on a website. Basically a vast amount of historical information accumulated is queried based on the current browsing behavior of a user to provide a personalized experience.

Breese *et al*. [1] identified two major classes of collaborative filtering algorithms: *memory-based* and *model-based*. Memory-based algorithms operate on the entire access dataset, which records previous site usage, in order to make predictions. These algorithms employ a notion of distance to find a set of users, known as *neighbors*, which tend to agree with the active user. The preferences of neighbors are then "combined" to produce a prediction or top-N recommendations for the active user. The model-based algorithms, on the other hand, use the access dataset to estimate or learn a model, which is then used for predictions. Web usage mining techniques such as clustering, association rule mining, and sequence pattern discovery have been used for this purpose [9]. These techniques extract characteristics (patterns) from the access dataset through an offline process and use these patterns to generate recommendations in the online process. Two important challenges in CF-based recommender systems are *accuracy* and *scalability*. Memory-based techniques are simple, provide high accuracy recommendations, and admit easy addition of new data. However, they are computationally expensive as the size of the input dataset increases. On the other hand, model-based techniques reduce the online processing cost. This often comes at the cost of reduced accuracy of recommendation results.

To overcome these difficulties, we proposed a technique in [19], which is a hybrid of memory-based and model-based CF approaches, inheriting the advantages of both. It is important to note that the term hybrid has been used in different senses in recommender systems earlier. Our sense of hybrid is to combine both memory-based and model-based approaches to provide more accurate predictions at reasonable cost. Our CF technique makes innovative use of the following two important properties of the model generated by Relational Fuzzy Subtractive Clustering (RFSC) [18], used in the offline learning phase. Firstly, the model has user sessions as cluster centers, giving us fuzzy cluster prototypes. Secondly, every session has varying degree of membership with each cluster. The fuzzy nearest prototype of the active user is used to find a group of like-minded users within which a memory-based search is carried out. A significant advantage of RFSC is that it scales to large datasets and does not require any user specified control parameters. Furthermore, we showed that RFSC is relatively more immune to noise. This is important in particular when dealing with web usage data which is inherently noisy in nature.

Another important aspect of web personalization is the dynamic nature of the usage of websites. A popular website is visited by a large number of users with a variety of needs. The browsing behavior of users is not fixed or static. A user might browse the same page for different purposes. Each time the user accesses the site, he/she may have different browsing goals. Moreover, user behavior and interest changes dynamically over a period of time. If these profiles or models do not incrementally adapt to the new interests and usage of the site, this static nature of modeling would lead to degradation of the system over time. Irrelevant and mistargeted recommendations annoy the customers leading to low impact of personalization systems. Most web usage modeling is static, mainly due to the fact that the time complexity to compile the data into a model can very often be prohibitive and adding one new data point may require a full recompilation of the model [15]. It is therefore important to develop maintenance schemes which can adapt these models to non-stationary environments.

In this paper, we focus on profile maintenance for a dynamically changing environment such as the web, which is one of the least developed areas of model-based CF. We present a new maintenance scheme, called incremental RFSC, as an extension of Relational Fuzzy Subtractive Clustering technique, that can track evolution of the clustering model. We study the similarity of the clustering model obtained by applying the maintenance algorithm and compare it with the model obtained from a complete reclustering to show the validity of our proposed maintenance algorithm. A maintenance technique is not an alternative for reclustering but enables adaptability and consecutive maintenance after reclustering. Any maintenance technique based on incremental update of the profile requires a measure to indicate, to the web analyst, when accumulated usage data has to be reclustered. For this we have proposed a quantitative measure, called *impact factor*. When the impact factor exceeds a predefined threshold, a remodeling is recommended. We conducted extensive experiments which compare the effectiveness of recommendations using our incremental RFSC technique, complete reclustering, and memory-based CF, for a reasonably large real dataset.

The organization of the rest of this paper is as follows. Section 2 reviews related work in cluster maintenance. In Section 3, we provide an overview of the Relational Fuzzy Subtractive Clustering algorithm. Our incremental RFSC algorithm for model maintenance is introduced in section 4. Section 5 reports the results of our experiments on synthetic as well as on a real web log dataset. Concluding remarks and future work are presented in Section 6.

## 2. OVERVIEW OF CLUSTER MAINTENANCE

The goal of clustering is to find "natural classes" in a set of given objects such that similar objects get grouped together in the same class. A clustering algorithm should have the following desirable properties [21, 2]:

(1) Stability -- producing a clustering which is unlikely to be altered drastically when new objects are added.

(2) Robustness -- small errors in the description of the objects may only lead to small changes in the clustering.

(3) Order insensitivity -- composition of clusters is independent of the order in which objects are processed.

(4) Maintainability -- handles growth efficiently, i.e., maintenance of clusters should be practical and efficient.

Most clustering algorithms are not suitable for maintaining clusters in a dynamic environment, and often the problem of updating clusters is not solved without a complete reclustering [21, 3, 2, 20]. One reason is that clustering and maintenance techniques are merely based on heuristics which very often make reclustering of the entire dataset more practical.

One of the early works done in dynamic cluster maintenance in information retrieval is [2], which proposed a cluster maintenance scheme based on the notion of cover coefficient. An incremental document clustering algorithm proposed in [3] attempts to maintain small diameter clusters as new points are inserted in the database. The authors also studied the dual clustering problem, where the clusters are of fixed diameter, and the goal is to minimize the number of clusters. Ester et al. [5] have proposed an extension of GDBSCAN algorithm for large updates in a data warehouse environment. Tasoulis et al. [20] have presented an extension of the k-windows algorithm that can discover clustering rules from a dataset in a dynamic environment. Their experimental results show that k-windows algorithm can effectively and efficiently identify the changes in the pattern structure. In web usage mining, Shahabi et al. [17] studied dynamic clustering as an approach to make the cluster model adaptive to short-term changes in user behavior. They argue that web usage mining systems should compromise between scalability and accuracy to be applicable to web sites with numerous visitors. They show that accuracy of dynamic clustering is low, though it helps achieve adaptability. Nasraoui et al. [12] first proposed the framework of mining evolving data streams through a new scalable clustering methodology, inspired from the natural immune system to be able to continuously learn and adapt to new incoming patterns. The Web server plays the role of the human body, and the incoming requests play the role of foreign antigens, bacteria, or viruses that need to be detected by the proposed immune-based clustering technique.

Our work in this paper differs from earlier maintenance algorithms in several ways, most notably, the following:

- Most of the aforementioned maintenance schemes are for crisp clustering. Our maintenance scheme is for RFSC, which is a fuzzy clustering algorithm. There has been little work reported is this area.
- We strongly believe that computation of similarity between the clustering obtained after application of any maintenance algorithm and complete reclustering is necessary to validate a maintenance scheme. Unlike in many of the above, in this work we show the validity of using incremental RFSC by computing similarity measures for our experiments.
- We have provided a quantitative definition of an "impact factor." Thresholding of this factor can indicate when a complete remodeling process is required.

## 3. RELATIONAL FUZZY SUBTRACTIVE CLUSTERING

For completeness, we briefly describe the RFSC algorithm in this section. Interested readers are referred to [18] for a detailed description along with experimental results showing its effectiveness in web usage profiling. Numerical relational data describes the set of objects to be clustered, less directly by giving a measurement of the dissimilarity (or similarity) between each pair of objects. Relational data [6] is presented as a matrix R, where $R_{ij}$ is the dissimilarity between objects $x_i$ and $x_j$. It also holds that $R_{ij} \geq 0$, $R_{ij} = R_{ji}$, and $R_{ii} = 0$. RFSC algorithm starts by considering each object as a potential cluster center. The potential of any object $x_i$ is calculated using the formula:

$$P_i = \sum_{j=1}^{N_U} e^{-\alpha R_{ij}^2} \text{ , where } \alpha = 4/\gamma^2$$

in which $R_{ij}$ is the dissimilarity between objects $x_i$ and $x_j$, $N_U$ is the total number of objects to be clustered, and $\gamma$ is essentially the neighborhood calculated from the relational matrix R. In [18] we proposed a heuristic method for estimating a value for $\gamma$ as follows. We define neighborhood-dissimilarity value ($\gamma_i$) of each object i from every other object as the median of dissimilarity values of object i to all other objects. The neighborhood-dissimilarity value ($\gamma$) for the entire dataset is defined as the median of all $\gamma_i$'s, for i in $[1..N_U]$. For RFSC algorithm, the relational matrix is required to be a dissimilarity matrix with $0 \leq R_{ij} \leq 1$. Hence it can be seen that $\gamma$ will always be a value in the range [0,1]. This is a heuristic that seems to work fine for many datasets which we have used in our experiments.

The object with the highest potential ($P_1^*$) is selected as the first cluster center. Next, the potential of each object is reduced proportional to the degree of similarity with this previous cluster center. Thus there is larger subtraction in potential of objects that are closer to this cluster center compared to those which are farther away. After this subtractive step, the object ($x_t$) with the next highest potential ($P_t$) is selected as the next candidate cluster center. Now to check whether this candidate cluster center can be accepted as an actual cluster center or should be rejected, we make use of two threshold values $\overline{\in}$ (accept ratio) and $\underline{\in}$ (reject ratio). Also, we have that $0 < \overline{\in}$, $\underline{\in} < 1$, and $\underline{\in} < \overline{\in}$. If $P_t > \overline{\in} P_1^*$, then $x_t$ is selected as the next cluster center, and this is followed by the subtractive step described above. If $P_t < \underline{\in} P_1^*$, then $x_t$ is rejected, and the clustering algorithm terminates. If the potential $P_t$ lies between $\overline{\in} P_1^*$ and $\underline{\in} P_1^*$, then we say that potential has fallen in the gray region, and we check if the object provides a good trade-off between having a sufficient potential and being sufficiently far from existing cluster centers. If this is the case, then it is selected as the next cluster center. This process of subtraction and selection continues until $P_t < \underline{\in} P_1^*$, which is the termination condition. After finding C cluster centers, we can find the membership of different $x_j$ with each cluster $c_i$ using the formula:

$$u_{ij} = e^{-\alpha R_{c_i j}^2} \text{ , } i = [1..C] \text{ \& } j = [1..N_U],$$

in which $R_{c_i j}$ is the dissimilarity of the ith cluster center $x_{c_i}$ with the jth session $x_j$. When $x_j = x_{c_i}$, we have $R_{c_i j} = 0$ and the membership $u_{ij} = 1$. We have relaxed the constraint which fuzzy C-means based algorithms impose, namely that $\sum_{i=1}^{C} u_{ij} = 1$. This effectively makes RFSC less sensitive to noise, as shown by our experimental results in [18].

### Cluster validity index for RFSC

Any clustering algorithm is incomplete without an accompanying definition of a cluster validity index that gives a measure of the "goodness" of the clustering. Good clustering of data results in low intra-cluster distance and large inter-cluster distance. In [18], we defined a goodness index for RFSC algorithm based on Xie-Beni index [12]. Xie- Beni index is basically the ratio of compactness to the separation of the clusters, but it requires to have $\sum_{i=1}^{C} u_{ij} = 1$, which is a condition not mandatory in RFSC.

We define *compactness* as:

$$Compactness = \sum_{i=1}^{C} \left( \frac{\sum_{j=1}^{N_U} u_{ij}^2 * R_{c_i j}^2}{\sum_{j=1}^{N} u_{ij}} \right) \Big/ C$$

where $c_i$ is the ith cluster center.

*Separation* is defined as $\min_{i \neq k} R_{c_i c_k}^2$, for i=1 to C and k=1 to C, where $c_i$ and $c_k$ are the ith and kth cluster centers respectively, and $R_{c_i c_k}$ is the dissimilarity between these cluster centers.

$$Index\ of\ goodness = \frac{Compactness}{Separation}$$

We obtain a best clustering by using those values of $\overline{\in}$ and $\underline{\in}$ which results in a minimum value for this index.

## 4. INCREMENTAL RELATIONAL FUZZY SUBTRACTIVE CLUSTERING

Database contents or web usage data grow dynamically. Though the clusters obtained from usage data manifest the interest and trends among the users accessing the site at the time the clustering was applied, the interests and needs of users change dynamically over time. Hence we require some cluster maintenance scheme by which these changing trends and patterns can be captured without having to apply frequently this relatively expensive operation of reclustering of large volume of old and new data together.

Cluster maintenance is not a complete alternative to reclustering. By its very definition, cluster maintenance tries to incorporate newly arrived data into the existing model, while maintaining the profiles created from the originally given set of data. It can therefore at best be a close approximation to clustering of the complete data, old plus new. Thus reclustering will always give more accurate results and will be required to be done periodically. Cluster maintenance however enables us to adapt to the dynamic and changing environment in a much less expensive manner in terms of computation times and resources and also enables consecutive maintenance even after reclustering. Thus an optimal combination of full data clustering and cluster maintenance is ideally suited for dynamic environments. In this section, we describe our incremental RFSC algorithm that is an extension of the RFSC algorithm and forms part of our usage profile maintenance scheme. Within our maintenance scheme, we also present a measure that enables us to determine the stage at which a complete reclustering is required.

Let us assume that we have a set of $N_U$ objects and C clusters. Also $u(C \times N_U)$ is the membership matrix which contains the membership values of each object to each of the C clusters. To begin with, these C clusters and the matrix *u* are obtained using RFSC algorithm. Subsequently, these are updated using incremental RFSC until reclustering of the complete dataset is required. These cycles of incremental RFSC and reclustering make up the maintenance scheme.

Let W= {$Z_1$, Z2,…, $Z_C$} be the set of C prototypes representing these C clusters. We also store the corresponding potentials at which these C prototypes were chosen as the cluster centers, P= {$P_1$, $P_2$,…, $P_C$}. These potentials have a natural ordering (descending) as RFSC finds the cluster with the highest potential first, followed by the next highest potential, and so on.

Let $x_{new}$ be the new object which is required to be inserted. Depending upon $x_{new}$ we have three possible situations:

Case i) $x_{new}$ has a high membership value with respect to one of the existing clusters and hence its potential is such that it becomes a core member of that cluster. Or it has a very low potential in which case it is probably a noise.

Case ii) The potential of $x_{new}$ is such that it is not a core member of any of the clusters, its potential is lower than all the potentials of the C clusters, but is high enough to become a new cluster center.

Case iii) The potential of $x_{new}$ is such that it is better than the potential of one of existing cluster centers.

The process of determining the case that is applicable to the new object and the actions taken are described below:

1. First, the stored potentials of all previous cluster prototypes $P_i$ are raised by the degree to which this $x_{new}$ is close to these previously found cluster prototypes. Hence the potential of the cluster prototype will increase by a larger degree if $x_{new}$ is closer to it (as $x_{new}$ makes that cluster more dense) than when $x_{new}$ is farther away. The maximum increase in potential is 1, i.e., when $x_{new}$ coincides with an existing cluster center. Let these updated potentials be $P'_i$.

2. We check if $x_{new}$ has potential better than the first cluster center, i.e., $P_{new} > P'_1$. If yes, then Case (iii) holds and the actions taken are described later below. If not, we proceed with the subtractive step, i.e., potential of $x_{new}$ will be reduced depending on its similarity with the first cluster center. After this subtractive step, if $P_{new}$ becomes negative, it means that $x_{new}$ is a core element or very close to the first cluster center (Case (i) is true). No further comparison or subtraction is required.

3. If the potential does not become negative, then we continue to compare the updated potential of $x_{new}$ with the potential of next cluster prototype. Again if the potential of $x_{new}$ is less than this cluster center, then we perform the subtractive step and the above discussion holds. If the potential of $x_{new}$ is greater than the current cluster center, i.e., $x_{new}$ has potential better than the potential of an existing cluster center (Case (iii) holds) then we continue with step 4a, otherwise we proceed with the subtractive step. Also it might so happen that after comparison with the potentials of all the existing C cluster prototypes and going through the corresponding subtractive steps $x_{new}$ may have a potential good enough to become a new cluster center (Case (ii) is true) then we continue with step 4b.

4a. If Case (iii) is true, we calculate the impact factor for not making $x_{new}$ as one of the existing cluster centers. Computation of impact factor is discussed below. If the impact factor exceeds a threshold, specified by a web administrator, then a complete reclustering has to be performed.

4b. If Case (ii) is true, $x_{new}$ becomes a new cluster centre and the model now has C+1 prototypes. Additionally, the membership of all the existing objects is also calculated with respect to this new cluster and stored in matrix u which will now have one more row to store the membership values for the new cluster.

5. Irrespective of which case is true, the membership of $x_{new}$ is calculated with all the existing clusters.

## Impact Factor

Impact factor, denoted $\mathcal{I}$, is a measure of the degree to which the cluster model is affected by not making the new incoming object as one of the existing cluster centers, when its potential is such that using RFSC (regular clustering of the complete dataset) it would have been a cluster centre. Impact factor is calculated only when case (iii) is true. We first explain the notion of impact factor with the help of an example and then provide a quantitative definition.

Fig. 1(a) shows a clustering of $N_U$ objects obtained from RFSC. There are three clusters and their cluster centers are shown. Now suppose a new object $x_{new}$ arrives and has to be incorporated in the existing clustering. If case (iii) becomes true, then. $x_{new}$ has potential better than one of the existing cluster centers. Two situations can arise. The first situation is as shown in Fig. 1(b), wherein we see that $x_{new}$ lies very close to an existing cluster center. Making $x_{new}$ the cluster center of this cluster will not change the clustering by much because there already exist a cluster center very close to $x_{new}$ and membership of $x_{new}$ will be very close to 1. Hence the impact of not making this a cluster center is very low. The other situation is the one shown in Fig. 1(c), wherein we see that $x_{new}$ is sufficiently far from any of the existing cluster centers and hence the impact of not making this object as a cluster center will be much greater.

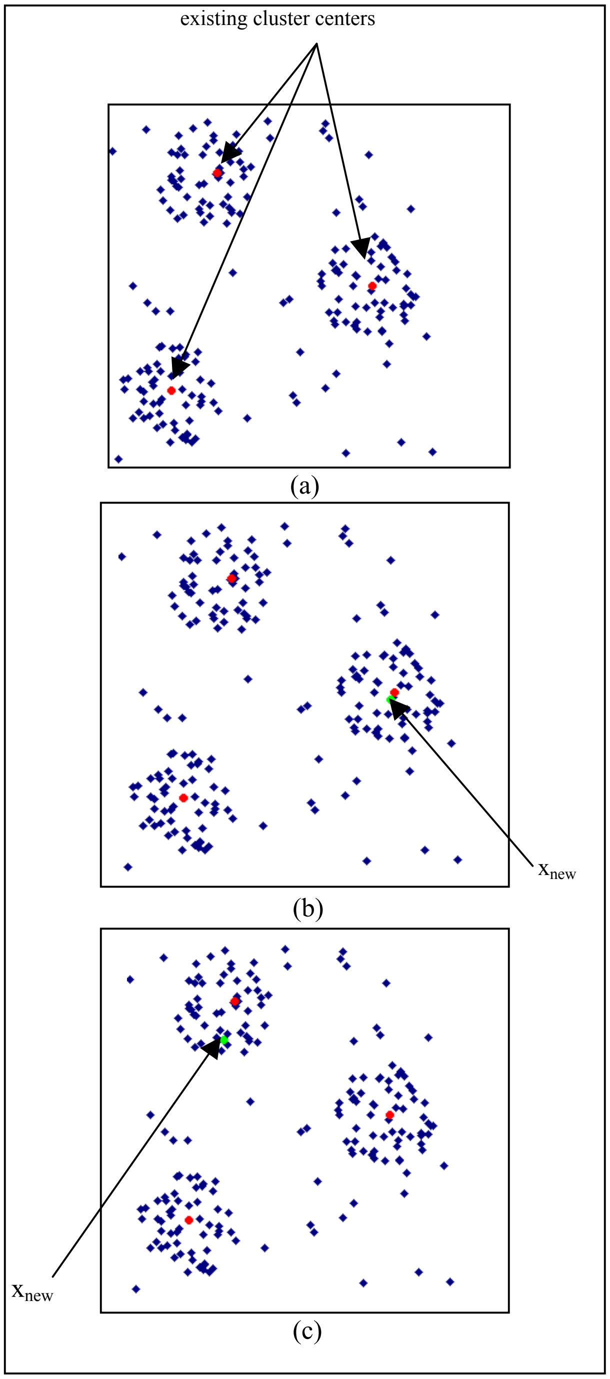


**Figure 1. Example for impact factor calculation.**

We shall now give a quantitative measure for impact factor. The impact factor $\mathcal{I}$ is initialized to 0 the first time RFSC algorithm is applied and then onwards every time the dataset is reclustered. When a new object $x_{new}$ is to be added to the current clustering, if Case (iii) holds, we find the minimum of the dissimilarity of $x_{new}$ to existing cluster centers and increment the impact factor. Hence if there is a cluster center near to $x_{new}$, the impact is less and if the clusters centers are far away, the impact is more. When the impact factor reaches a pre-defined threshold,

reclustering of the entire dataset is required. Currently we are working with a threshold that is a multiple of γ.
The maintenance scheme with the impact factor considered is presented as follows.

### Incremental RFSC Algorithm

```
1) Calculate the potential $P_{new}$ of the object to be inserted $x_{new}$
2) Raise the potentials of all existing cluster prototypes;
        $D_{i,new}$ = dissimilarity between $x_{new}$ and $i^{th}$ cluster center
        $P_i = P_i + e^{-\alpha D_{i,new}^2}$;
3) $d_{min}$ = minimum of the dissimilarity values between $x_{new}$ and all
the previously found cluster centers.
    for i=1 to C do
        if ($P_{new} > P_i$) then
            if ($P_{new} > \bar{\in} P_1$)        //case (iii) true
                $I$ += $d_{min}$
            else if ($P_{new} \geq \underline{\in} P_1$)
                If ($d_{min}/\gamma$)+($P_{new}/P_1$) ≥ 1, then
                //case (iii) is true
                    $I$ += $d_{min}$
                end if
            end if
        else
            $P_{new} = P_{new} - P_i\, e^{-\alpha D_{i,new}^2}$;
            If ($P_{new} < 0$) goto step 5;
    // no further comparisons required as case (i) is true
        end if
    end for
4) if (($P_{new}$ >0) and (Case (iii) is not true))
        //check if $x_{new}$ can become a new cluster center
        if ($P_{new} \geq \underline{\in} P_1$)
            If ($d_{min}/\gamma$) +($P_{new}/P_1$) ≥ 1, then
            //case (ii) is true, $x_{new}$ is the new cluster center
                C = C+1; //increment cluster count
                $P_C = P_{new}$;
                Calculate the membership of $N_U$ old
                objects with respect to this new
                cluster;
            end if
        end if
    end if
5) Increment $N_U$ by 1 as a new object is added;
    Calculate membership of $x_{new}$ with C clusters.
```

## 5. EXPERIMENTAL EVALUATION

In this section, we first present the evaluation metrics used by us for measuring the similarity between clustering obtained by applying maintenance algorithm and complete reclustering. This is followed by an analysis of the results of our experiments and the values obtained for these metrics.

### 5.1 Evaluation Metrics

For similarity analysis, we have used the Rand, Corrected Rand, and Jaccard coefficients. We measure the similarity between clusters generated by incremental RFSC and those generated by reclustering using RFSC. Below, we briefly describe these metrics [8].

Let $V = \{v_1, v_2, \dots, v_R\}$ and $W = \{w_1, w_2, \dots, w_C\}$ be two crisp partitions of *n* objects, that is, there are R clusters in partition *V* and C clusters in partition *W.* We construct a contingency table for the two partitions. Entry $n_{ij}$ in Table 1 is the number of objects that are both in cluster $v_i$ and cluster $w_j$. The value $n_{i.}$ is the sum of values in row i, or the number of objects in cluster $v_i$, and $n_{.j}$ is the number of objects in cluster $w_j$.
Considering each pair of objects $x_i$ and $x_j$, there are four types of possibilities to consider, described as follows.

Type 1: $x_i$ and $x_j$ belong to the same cluster $v_a$ of V, and to the same cluster $v_b$ of W.

Type 2: $x_i$ and $x_j$ belong to different clusters of V but to the same cluster of W.

Type 3: $x_i$ and $x_j$ belong to the same cluster of V but to different clusters of W.

Type 4: $x_i$ and $x_j$ belong to different clusters of V and to different clusters of W.

**Table 1 Contingency Table for Partitions *V* and *W***

| | | | | | |
|---|---|---|---|---|---|
| | $w_1$ | $w_2$ | … | $w_C$ | |
| $v_1$ | $n_{11}$ | $n_{12}$ | … | $n_{1C}$ | $n_{1.}$ |
| $v_2$ | $n_{21}$ | $n_{22}$ | … | $n_{2C}$ | $n_{2.}$ |
| . | | | | | . |
| . | | | | | . |
| . | | | | | |
| $v_R$ | $n_{R1}$ | $n_{R2}$ | … | $n_{RC}$ | $n_{R.}$ |
| | $n_{.1}$ | $n_{.2}$ | | $n_{.C}$ | |

Suppose the frequencies of these four types are *a, b, c, d*, respectively. Then we have:

$$a + b + c + d = \frac{n(n-1)}{2}$$

These frequencies can be calculated from the contingency table as follows:

$$a = \sum_i \sum_j \binom{n_{ij}}{2}; \qquad b = \sum_j \genfrac{}{}{0pt}{}{n_{.j}}{2} - \sum_i \sum_j \binom{n_{ij}}{2};$$

$$c = \sum_i \frac{n_{i.}}{2} - \sum_i \sum_j \binom{n_{ij}}{2}; \qquad d = \binom{n}{2} - a - b - c;$$

Using these values, we have that:

Rand index = $(a + d) / \binom{n}{2}$ and

Jaccard coefficient = $a / (a + b + c)$.

Rand and Jaccard values are in the range [0, 1]. The maximum value of 1 may not be achievable when the two partitions have different number of clusters. Hubert *et al.* [7] suggested to using rand index corrected for chance, called as "corrected" Rand (CR) coefficient. Correcting a statistic for chance means normalization such that its value is 0 when the partitions are selected by chance, and is 1 when a perfect match is achieved. It is defined as the following formula:

CR=

$$\left( \frac{\sum_i \sum_j \binom{n_{ij}}{2} - \left[ 1 \Big/ \binom{n}{2} \right] \sum_i \binom{n_{i.}}{2} \sum_j \binom{n_{.j}}{2}}{\left(1/2\right) \left[ \sum_j \binom{n_{.j}}{2} + \sum_i \binom{n_{i.}}{2} \right] - \left[ 1 \Big/ \binom{n}{2} \right] \sum_i \binom{n_{i.}}{2} \sum_j \binom{n_{.j}}{2}} \right)$$

## 5.2 Experiments and Results

For our experiments, we have used the user access logs from the web server of Computer Science and Software Engineering Department (CSE) at Concordia University. The access log records from a web server are at a very fine granularity. Cleaning [4, 9] is required to remove log entries for image files and other such components within a web page. Log records for accesses by web crawlers and failed requests are also removed from the dataset. A *session* is typically the collection of URLs of pages visited by a user from the moment the user enters a web site to the moment the same user leaves it. Each distinct URL in the site is assigned a unique number j ranging from 1 to M, the total number of URLs. The collection of the user accesses in the kth session is represented as a binary vector of size M, in which the jth entry is 1 if the user accessed the jth URL during this session, and is 0 otherwise. Sessions were identified from the log records by considering 45 minutes as the maximum elapsed time between two consecutive accesses from a single IP address. Root "/" was filtered out from all sessions as it appeared in more than 80% of the sessions. We also removed short sessions of length 1 or 2, as they do not carry much information related to users' access patterns. After this pre-processing phase, we obtained 12,227 user sessions with 10,153 distinct URLs. The average length of the sessions was 8.54 pages and sparsity level was 0.999312, defined as 1- (nonzero entries / total entries).

We use the similarity measure proposed in [13, 14] for construction of relational matrix R. Session similarity in turn is defined based on URL similarity. The syntactic similarity between the ith and jth URLs is defined as follows:

$$S_u(i,j) = \min\left(1, \frac{|p_i \cap p_j|}{\max(1, \max(|p_i|, |p_j|) - 1)}\right),$$

where pi denotes the path traversed from the root node (the main page) to the node which corresponds to the ith URL. We use |pi| to denote the length of path pi.

In defining similarity of any two sessions sk and sl, two measures may be used. The first measure is cosine, which does not consider the site structure. It is defined as follows:

$$S_{1,kl} = \sum_{i=1}^{M} s_{ki} s_{li} \Big/ \sqrt{\sum_{i=1}^{M} s_{ki} \sum_{i=1}^{M} s_{li}}$$

The second similarity measure, defined below, incorporates syntactic URL similarity.

$$S_{2,kl} = \sum_{i=1}^{M} \sum_{j=1}^{M} s_{ki} s_{lj} S_u(i,j) \Big/ \sum_{i=1}^{M} s_{ki} \sum_{j=1}^{M} s_{lj}$$

The similarity between any pair of sessions *k* and *l* is a value in [0,1], defined as $S_{kl} = \max(S_{1,kl}, S_{2,k})$. It follows that the dissimilarity between sessions *k* and *l* is $D_{kl} = 1 - S_{kl}$, which is a non-Euclidean similarity measure. Note that the dissimilarity matrix R is non-Euclidean in nature.

### 5.1.1 Similarity Analysis of Incremental RFSC

Let $N_{old}$ denote the number of sessions in the initial set of sessions, and $N_{new}$ denote the number of new sessions that are required to be added to the cluster model of the $N_{old}$ sessions. We created six experimental case studies by choosing different values of $N_{old}$ as 7000 (57.25%), 8000 (65.43%), 9000 (73.61%), 10000 (81.79%), 11000 (89.96%), and 12000 (98.14%). For each case, the number of increment $N_{new}$ = 12227 - $N_{old}$. These divisions of the original dataset of into old and new sessions were done randomly.

For each case, we applied RFSC to cluster the $N_{old}$ sessions and used the validity index mentioned above to achieve the best clustering for these $N_{old}$ sessions. We then applied our maintenance algorithm in each case to add the new $N_{new}$ sessions to the existing clustering. For validation and comparison purposes, we also performed reclustering of the entire dataset, i.e., $N_{old} + N_{new}$ = 12227 sessions, using validity index to achieve the best clustering for the entire dataset. Since Rand, CR, and Jaccard coefficients are defined for crisp clustering, we defuzzified the clusters obtained by applying the maintenance and reclustering techniques. Each session is assigned to a cluster to which its degree of membership is the highest (nearest prototype). Also noise sessions which have low membership with all the prototypes are put into an extra cluster, called as *noise* cluster. That is, if a clustering yields C clusters, we add the (C+1)th cluster as the noise cluster during the defuzzication step. After defuzzification, we measure the similarity in clustering obtained from the incremental RFSC and reclustering operations. The values are shown in Fig. 2.

Particularly interesting is the Corrected Rand (CR) coefficient since it has the maximum value of 1 when the clustering structures (partitions) compared are identical, and a value 0 when they are generated by chance (i.e., CR is corrected for chance).

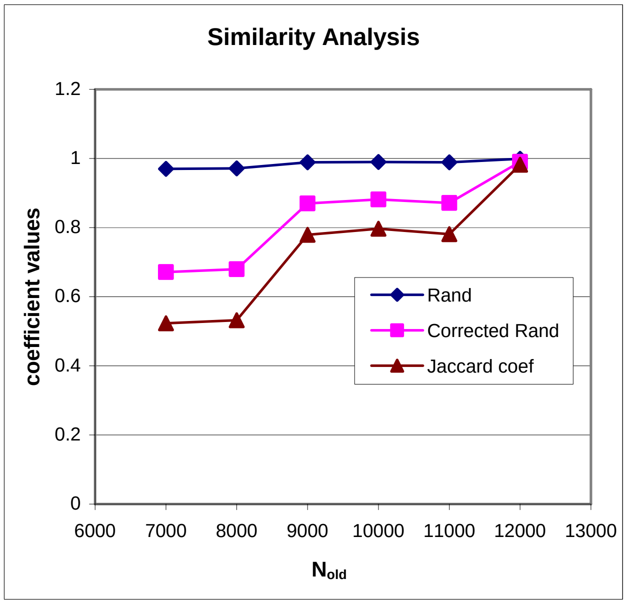


**Figure 2. Similarity between incremental RFSC and reclustering**

The CR values obtained are 0.671, 0.679, 0.87, 0.881, 0.871, and 0.99, respectively for the six cases studied in our experiments. We see that increase in $N_{old}$ or decrease in the number of newly added sessions $N_{new}$ leads to higher similarity in clustering. We also see that high values of CR are obtained even for low values of $N_{old}$. These results indicate that the similarity between the clustering obtained from incremental RFSC and complete reclustering did not, in a statistical sense, happen by chance. High values of CR for very small increments (0.99 for case 6 with $N_{old}$=12000) indicate that RFSC is stable when the dataset grows, which is a desirable clustering property as discussed in Section 2.

We also carried out some experiments using several synthetically generated datasets consisting of points in two-dimensional Euclidean space. One such dataset, shown in Fig. 3, contains 400 points. This dataset is randomly generated and has three prominent clusters among the noises.

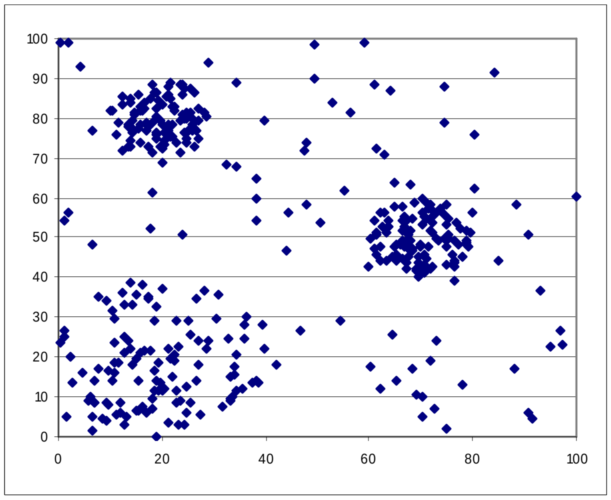


**Figure 3. Synthetic dataset with three clusters and noise**

As we can see, this dataset contains three natural clusters centered approximately at around (20,20), (70,50), and (20,80). Intentionally, we kept the density of the two clusters centered at (70, 50) and (20,80) the same, and double that of the cluster centered at (20,20). Normalized Euclidean distances were computed to define the dissimilarity matrix R. We then applied RFSC algorithm and used validity index to determine the best clustering. We obtained the least value of the index at C=3. The clusters centers found (in the order of their potentials) were (67.98, 49.25), (21.33, 78.23), and (19.37, 18.43). Here we remark that these are approximate cluster centers as the RFSC considers each data point itself as a potential cluster center. This makes sense in many applications, such as usage profiling through web log data, in which we are not interested in finding exact cluster centers but rather the usage profiles.

We next created a different kind of experimental case. We removed 100 points from the cluster centered at (20,80) which was found to be second most prominent cluster. These left us with a dataset of 300 points with two clusters and $\gamma$ = 0.37. We fix threshold for impact factor as 5*$\gamma$. We clustered these 300 points, using RFSC and the validity index, and found the best clustering at C=2. As a next step, we used our incremental RFSC maintenance algorithm to add the 100 points which were removed. We observed that the maintenance algorithm adds a new cluster with center at (20.39, 73.33), very close to (21.33, 78.23), which was found by reclustering of the entire dataset. Also the impact factor $\mathcal{I}$ was found to be 1.4374, which is quite low considering the fact that 100 new points were added to a dataset of 300 points. This also indicates that in most of the situations where Case (iii) was true, the candidate center was very close to an existing cluster center. Furthermore, $\mathcal{I}$ is less than the threshold 5*$\gamma$, indicating that remodeling is not required as yet.

### *5.1.2 Experiments to Examine Recommendation Quality*

The main goal of adapting the model to the dynamic changes in the environment is to increase the relevance and accuracy of recommendations to the user. We now discuss the results from some of the experiments performed to determine the recommendation quality when maintenance algorithm was used. Incremental RFSC fits in well with our fuzzy hybrid CF technique [19] and hence we compare recommendation quality for the following three techniques:

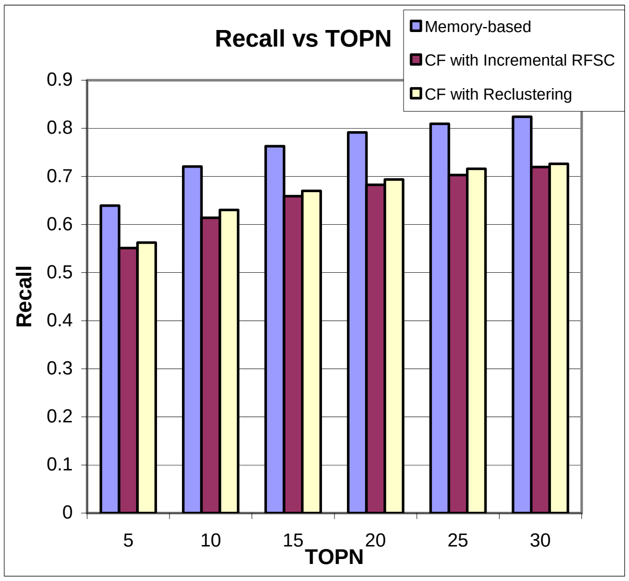
(i) fuzzy hybrid CF using incremental RFSC

(ii) fuzzy hybrid CF using Reclustering

(iii) Memory-base CF

In our earlier work [19], we showed that even if memory-based CF has slightly higher accuracy, efficiency of fuzzy hybrid CF is much better (30 to 40 times) than memory-based CF.

For this experiment, we randomly divided the dataset of user sessions into (1) the training set, with 8,000 sessions (65.43%) and (2) the test set, with 4227 sessions (34.57%). We applied the RFSC algorithm and the cluster validity index. In all, 33 clusters were found, i.e., C=33. We adopted the all-but-1 protocol used in [1] by which one page is randomly withheld (hidden) from every session in the test set. We then applied each of the three algorithms, memory-based (k-nearest neighbor), fuzzy hybrid CF technique using incremental RFSC, and fuzzy hybrid CF technique using reclustering to get a set of recommendations for every session in the test set. Finally, we checked to see if the hidden page was present in the top-N pages recommended by the respective algorithms. For fuzzy hybrid CF with incremental RFSC, after every 500 new sessions that were seen from the test set, these sessions were added to the clustering using incremental RFSC. Similarly for fuzzy hybrid CF with reclustering, complete reclustering was performed in steps of 500 sessions, i.e., at 8500, 9000, 9500, 10000, 10500, 11000, 11500, and 12000. Fig. 4 shows the comparison of recommendation effectiveness, using recall, precision, and F1 which have been widely used for measuring effectiveness [1, 15]. For this experiment, we kept the parameter k-nearest neighbor, k=100; fuzzy K-nearest prototype, K=4; and varied the parameter TOPN from 5 to 30.



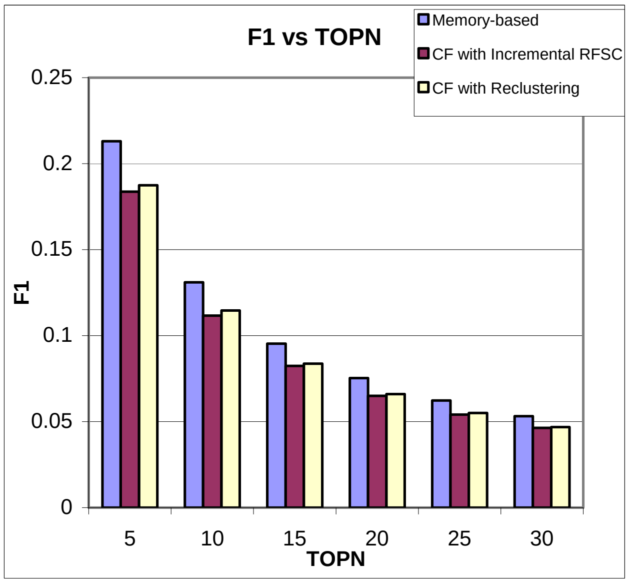


**Figure 4. Comparison of recommendation effectiveness**

We saw that recommendation quality obtained using incremental RFSC and complete reclustering is almost the same. Also as TOPN increases, even though we see an increase in recall, we see that precision decreases and the overall combined effect (captured by F1 measure) is a decreases as well.

## 6. CONCLUSION

The browsing behavior of users on the web is not fixed or static; rather, users' interests change dynamically with time. If models or profiles learned from usage data do not adapt according to the new interest and usage of the site, this static nature of modeling would lead to the degradation of the system over time. As this has to be online, the efficiency of this adaptation is crucial for the success of personalization. In this paper, we have proposed a new algorithm, incremental RFSC for dynamic usage profiling. We have experimentally validated the effectiveness of this cluster maintenance scheme by computing similarity of the resulting clustering with results from complete reclustering. These experiments show the similarity is high even for large number of new additions. We remark that incremental RFSC is not an alternative to reclustering. Periodically, complete reclustering will be required, depending on the growth rate and changes in users' browsing behaviors. We therefore introduced the computation of an impact factor which gives an idea as to when reclustering of the entire dataset is necessary. Most importantly, our experiments to determine the quality of recommendation obtained by applying incremental RFSC clearly indicate that the quality obtained is almost as good as complete reclustering of the entire dataset. As future work, we plan to experimentally investigate thresholds for impact factors in different dynamic situations. Also we would like to apply this incremental clustering algorithm in different domains like document clustering.

## 7. ACKNOWLEDGMENTS

This work was supported in part by Natural Sciences and Engineering Research Council (NSERC) of Canada and by ENCS Faculty Research, Concordia University. We also thank anonymous reviewers for their useful comments.

## 8. REFERENCES

[1] Breese, J., Heckerman, D., Kadie, C. Empirical analysis of predictive algorithms for collaborative filtering. In *Proc. of UAI-98*, pp. 43-52, 1998.

[2] Can, F., Ozkarahan, E.A. A dynamic cluster maintenance system for information retrieval. In *Proc. of the 10th Annual International ACM-SIGIR Conference*, pp. 123-131, 1987.

[3] Charikar, M., Chekuri, C., Feder, T., Motwani, R., 2004. Incremental clustering and dynamic information retrieval. In SIAM Journal on Computing 33 (6), pp. 1417-1440, 2004.

[4] Cooley, R., Mobasher, B. and Srivastava, J. Data Preparation for Mining World Wide Web Browsing Patterns. *Journal of Knowledge and Information Systems*, 1, pp. 1-27, 1999.

[5] Ester, M., Kriegel, H., Sander, J., Wimmer, M., Xu, X. Incremental clustering for mining in a data warehousing environment. In *Proc. of VLDB 1998*, Morgan Kaufmann Publishers Inc., pp. 323-333, 1998.

[6] Hathaway, R.J., Bezdek, J.C., Davenport, J.W. On relational data version of c-means algorithm. *Pattern Recognition Letters*, 17, pp. 607-612, 1996.

[7] Hubert, L. & Arabie, P. Comparing partitions, *Journal of Classification*, 2, pp. 193-198, 1985.

[8] Jain, A. K., Dubes, R. C. Algorithms for Clustering Data. Prentice-Hall, Englewood Cliffs, N. J., 1988.

[9] Mobasher, B. Web Usage Mining and Personalization. *Practical Handbook of Internet Computing*, Munindar P. Singh (ed.), CRC Press, 2004.

[10] Mobasher, B., Cooley, R., Srivastava, J. Automatic personalization based on web usage mining. *Comm. ACM*, 43, 8 (August), 142–151, 2000.

[11] Nasraoui, O. World Wide Web Personalization. *Encyclopedia of Data Mining and Data Warehousing*, J. Wang, (ed.), Idea Group, 2005.

[12] Nasraoui O., Cardona C., Rojas C., and Gonzalez F. Mining Evolving User Profiles in Noisy Web Clickstream Data with a Scalable Immune System Clustering Algorithm. In *Proc. of WebKDD*, Washington DC, August 2003.

[13] Nasraoui O., Krishnapuram R., Joshi A., Kamdar T. Automatic Web User Profiling and Personalization using Robust Fuzzy Relational Clustering. In *E-Commerce and Intelligent Methods*, Springer-Verlag, 2002.

[14] Nasraoui O., Frigui H., Krishnapuram R., Joshi A. Extracting Web User Profiles Using Relational Competitive Fuzzy Clustering. *International Journal on Artificial Intelligence Tools*, Vol. 9, No. 4, pp. 509-526, 2000

[15] Pennock, D. M., Horvitz, E., Lawrence, S., Giles, C. L. Collaborative filtering by personality diagnosis: A hybrid memory- and model-based approach. In *Proc. of UAI-2000*, pp. 473-480, Stanford, CA, 2000.

[16] Sarwar, B. M., Karypis, G., Konstan, J. A., Riedl, J. Analysis of recommender algorithms for e-commerce. In *Proc. of the 2nd ACM E-commerce Conference*, Minnesota, USA, 2000.

[17] Shahabi, C., Banaei-Kashani, F., A Framework for Efficient and Anonymous Web Usage Mining Based on Client-Side Tracking. In *Proc. of WEBKDD 2001*, Springer-Verlag, New York, 2002.

[18] Suryavanshi, B.S., Shiri, N., Mudur, S.P. An Efficient Technique for Mining Usage Profiles using Relational Fuzzy Subtractive Clustering. In *Proc. of* IEEE Int'l Workshop on Challenges in Web Information Retrieval and Integration *(WIRI' 05)*, Tokyo, Japan, April 8-9, 2005.

[19] Suryavanshi, B.S., Shiri, N., Mudur, S.P. A Fuzzy Hybrid Collaborative Filtering Technique for Web Personalization. In *Proc. of 3rd Workshop on Intelligent Techniques for Web Personalization (ITWP'05)*, Edinburgh, Scotland, August, 2005.

[20] Tasoulis, D., Vrahatis, M. Unsupervised Clustering on Dynamic Databases. *Pattern Recognition Letters* (to appear), 2005.

[21] Van Rijsbergen, C. J. Information Retrieval. 2nd ed. Butterworths, London, 1979.

[22] Xie, X. L., Beni, G. A validity measure for fuzzy clustering. *IEEE Trans. on PAMI*, 13(8), pp. 841-847, 1991.